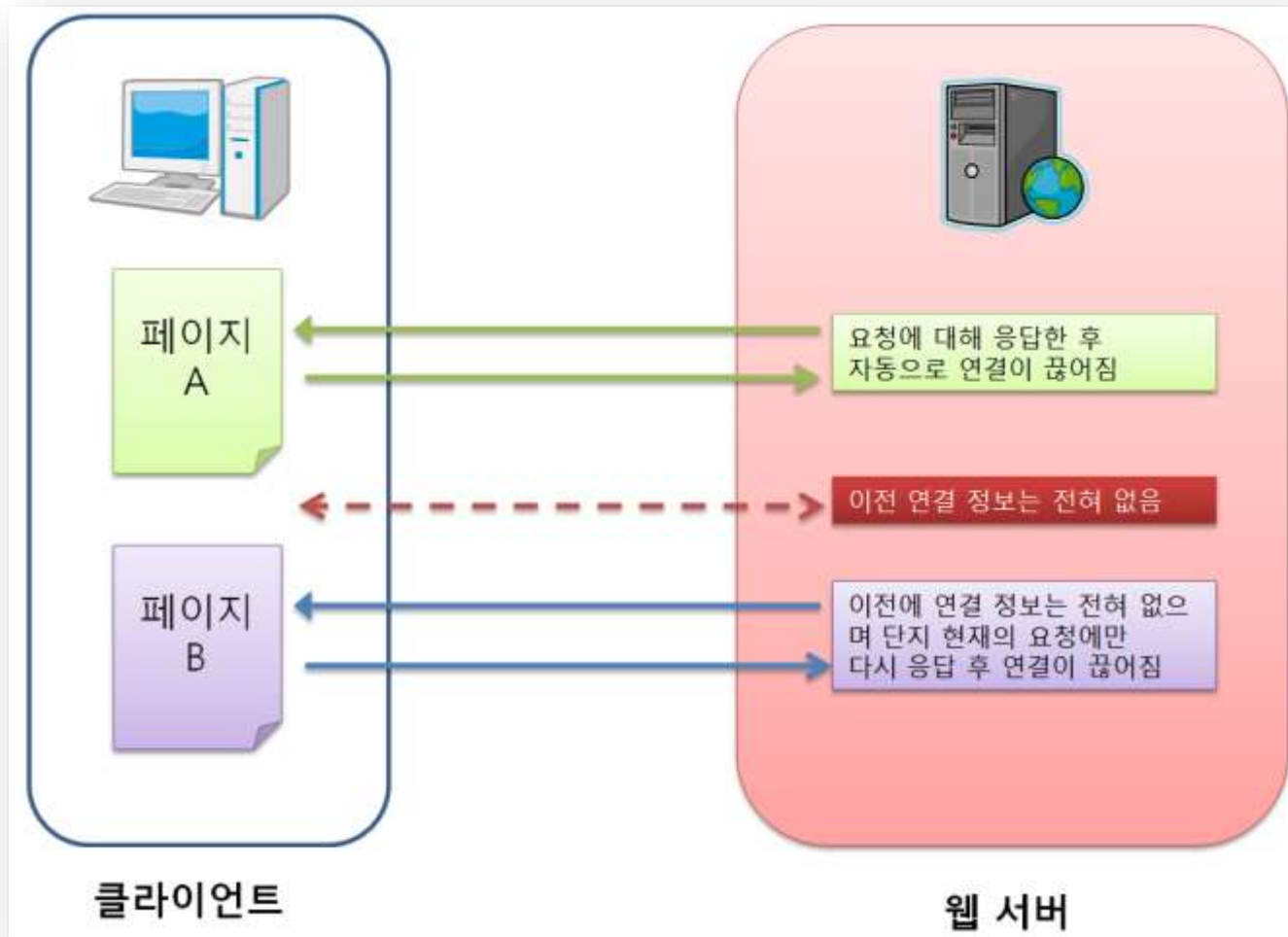


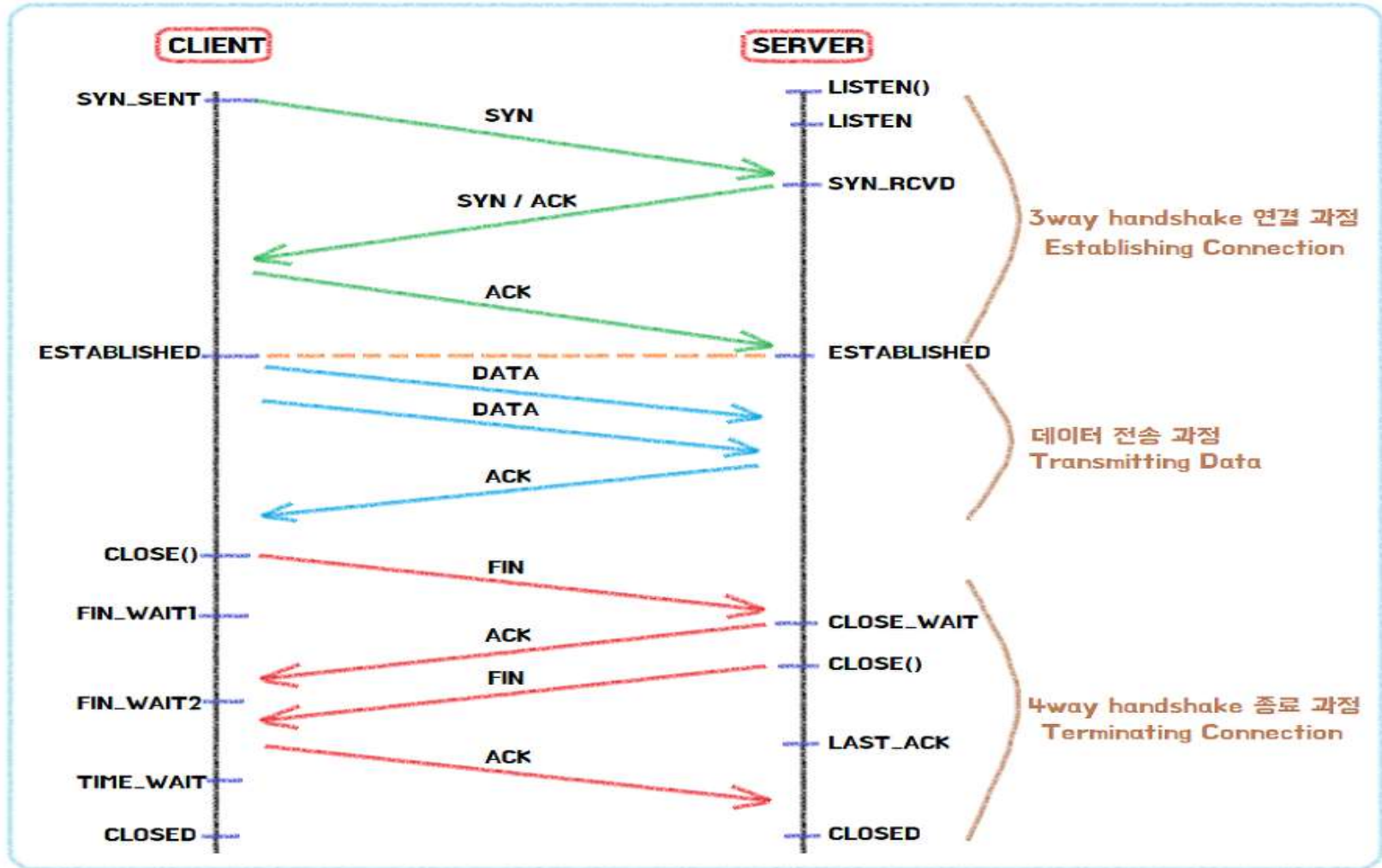
제 07 장 쿠키와 세션

2020년도 1학기

❖ Connectionless



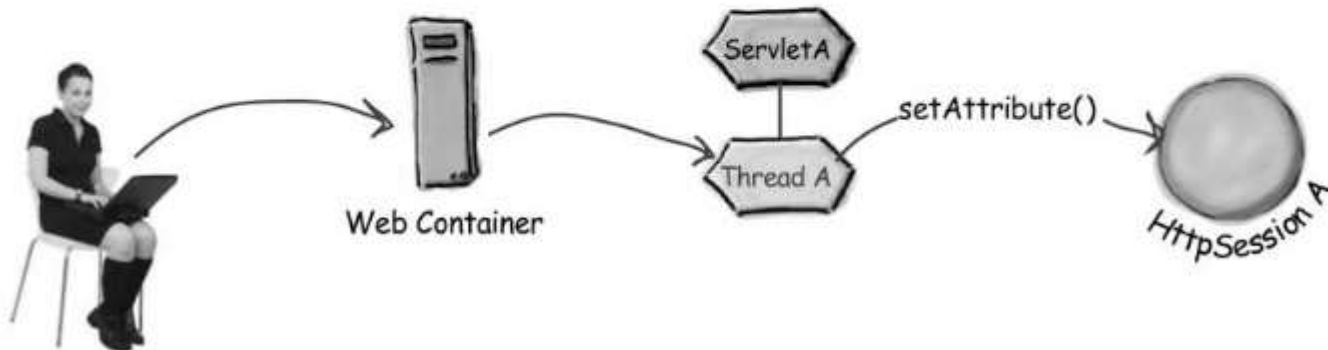
TCP 연결설정 특성





일단 아래와 같은 작업의 흐름을 봅시다~

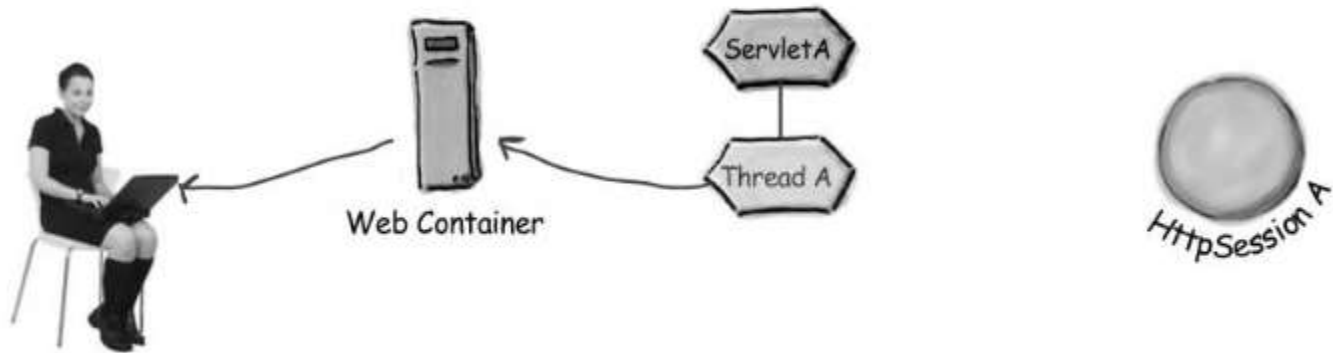
- ① 다이애나가 “흑맥주” 를 선택하고 보내기 버튼을 누름
- 컨테이너는 이 요청을 맥주주문처리 서블릿의 새로운 쓰레드에 전달함
- 맥주주문처리 서블릿의 쓰레드는 다이애나와 관련된 세션을 찾아서, 그녀의 선택사항인 “흑맥주” 를 세션 객체 내의 속성 필드에 저장함.





②

앞서의 서블릿은 비즈니스 로직을 처리한 후
응답메시지를 되돌려줍니다. 이 경우에는 후속 질문,
예를 들어, “가격대는 얼마를 원하세요?” 같은 걸 보낼
수 있을 겁니다.



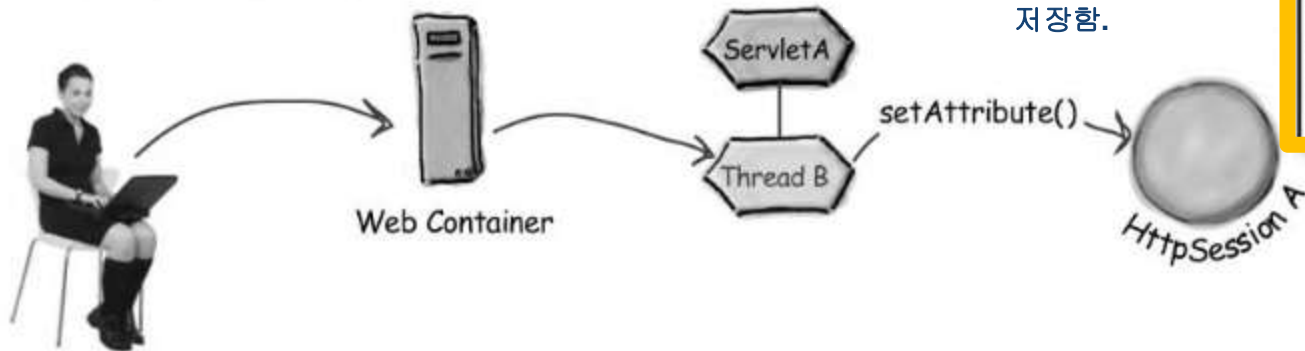


③ 다이애나는 표시된 페이지의 새로운 질문을 보고, “값비싼 것”을 선택하고 보내기 버튼을 누름

컨테이너는 이 요청을 맥주주문처리 서블릿의 새로운 쓰레드에다 전달함

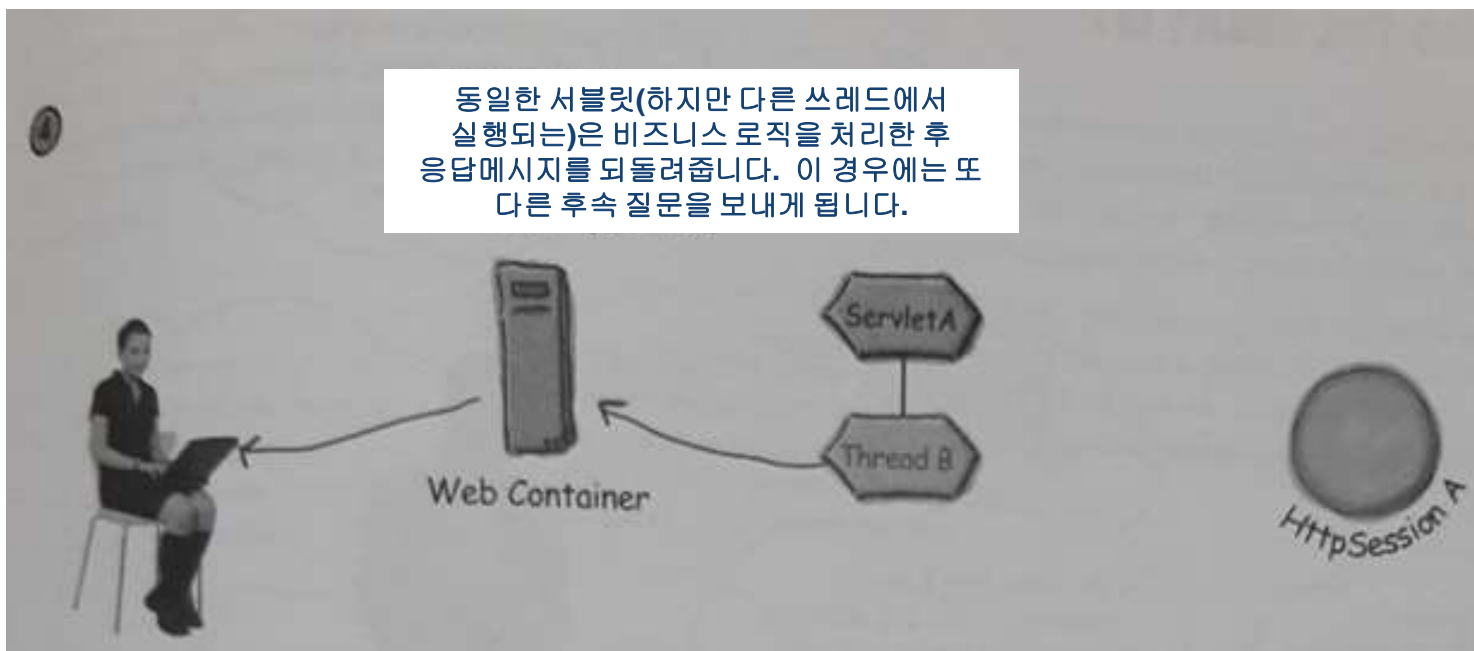
맥주주문처리 서블릿의 쓰레드는 다이애나와 관련된 세션을 찾아서, 그녀의 새로운 선택사항인 “값비싼 것”을 세션 객체 내의 속성 필드에 저장함.

Same client
Same **servlet**
Different request
Different thread
Same session



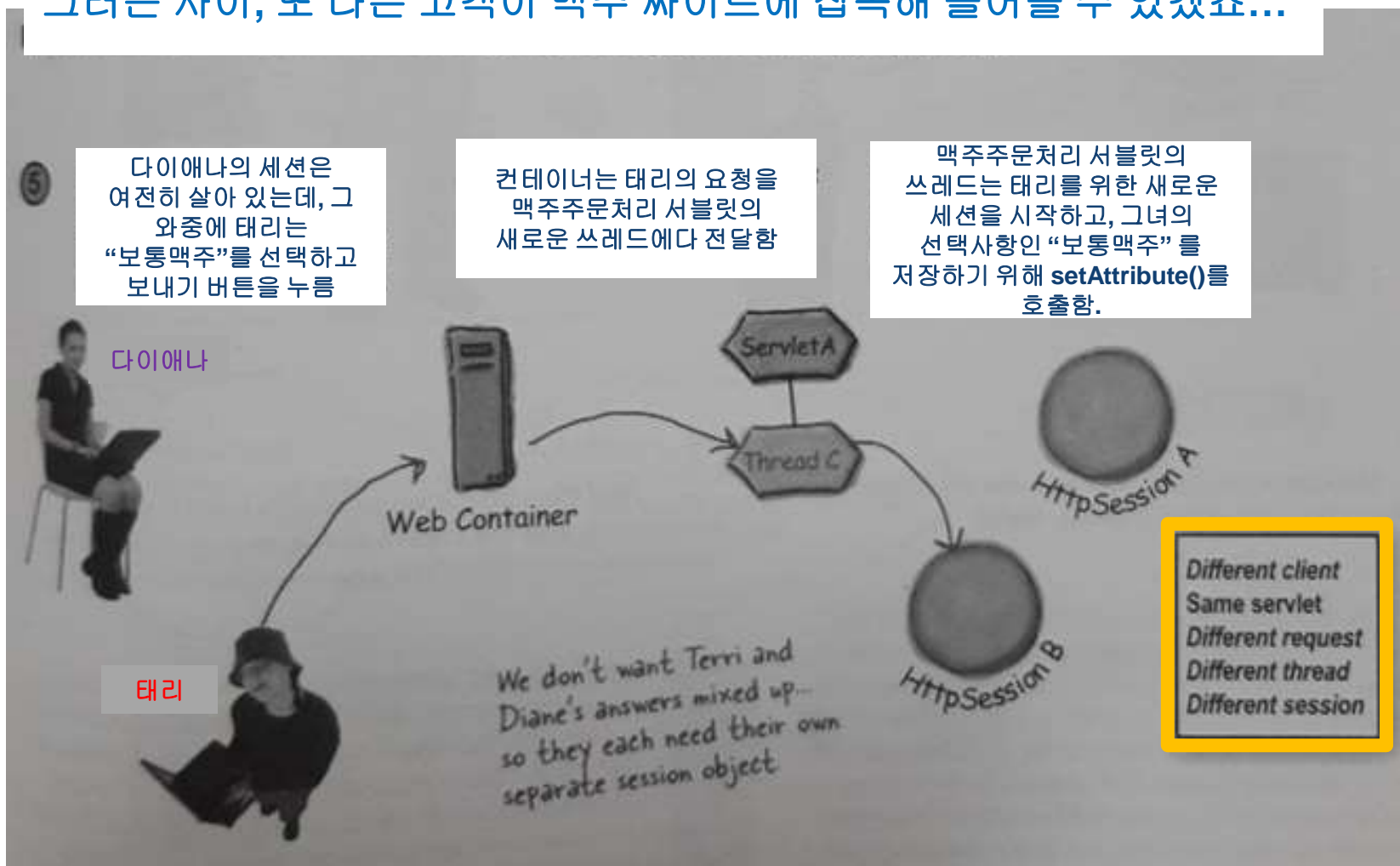


동일한 서블릿(하지만 다른 쓰레드에서 실행되는)은 비즈니스 로직을 처리한 후 응답메시지를 되돌려줍니다. 이 경우에는 또 다른 후속 질문을 보내게 됩니다.





그러는 사이, 또 다른 고객이 맥주 사이트에 접속해 들어올 수 있겠죠...



아이쿠 큰 문제가 있네요 ...



컨테이너가 어떻게
클라이언트들을 구분할 수
있을까요?

우리 서로 사랑하는 사이
아니었어? 그런 줄
알았는데...ㅠ



미안. 하지만 난 너를
모르겠어. 물론 우리
관계가 좋았을 거라
생각해. 다시 시작해보자.



컨테이너는 어떻게
다이앤과 태리를 식별할
수 있을까요? 모든 요청은
새로운 연결일텐데..

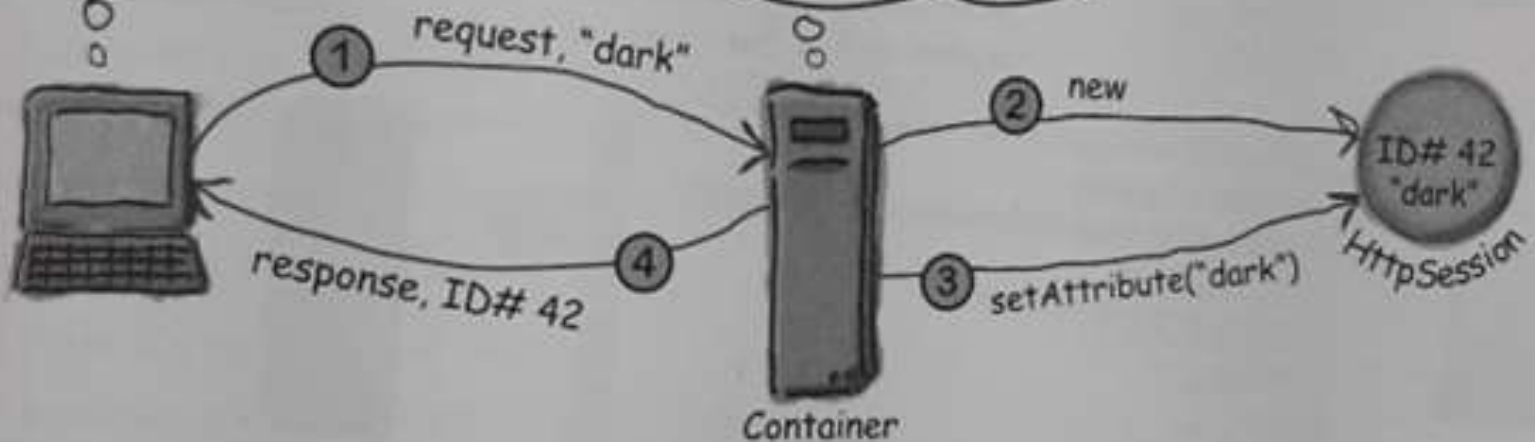


고유한 세션ID가 그 해결책!

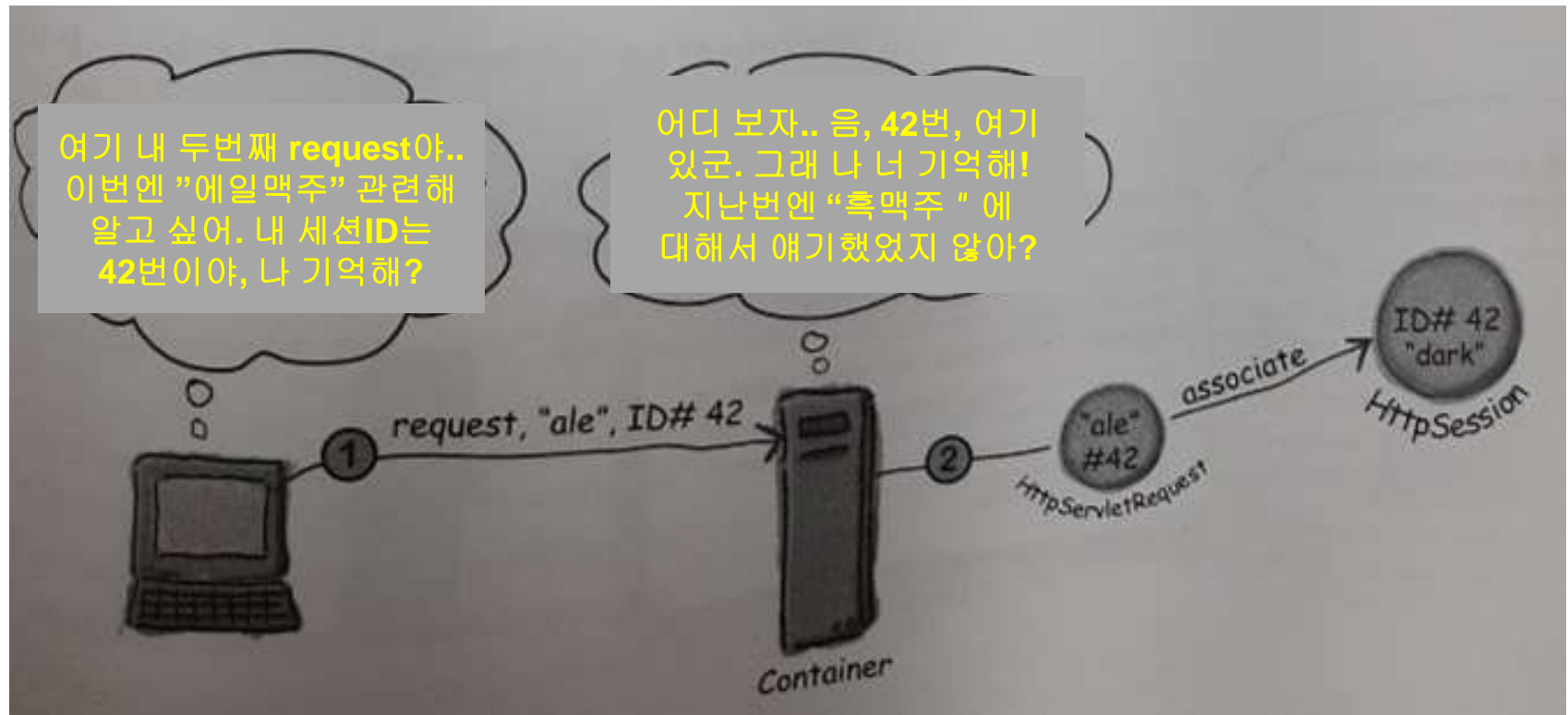


안녕 서버! 이번이 내
첫번째
request야.. "흑맥주"
관련해 알고 싶어. 우리
얘기 좀 할 수 있을까?

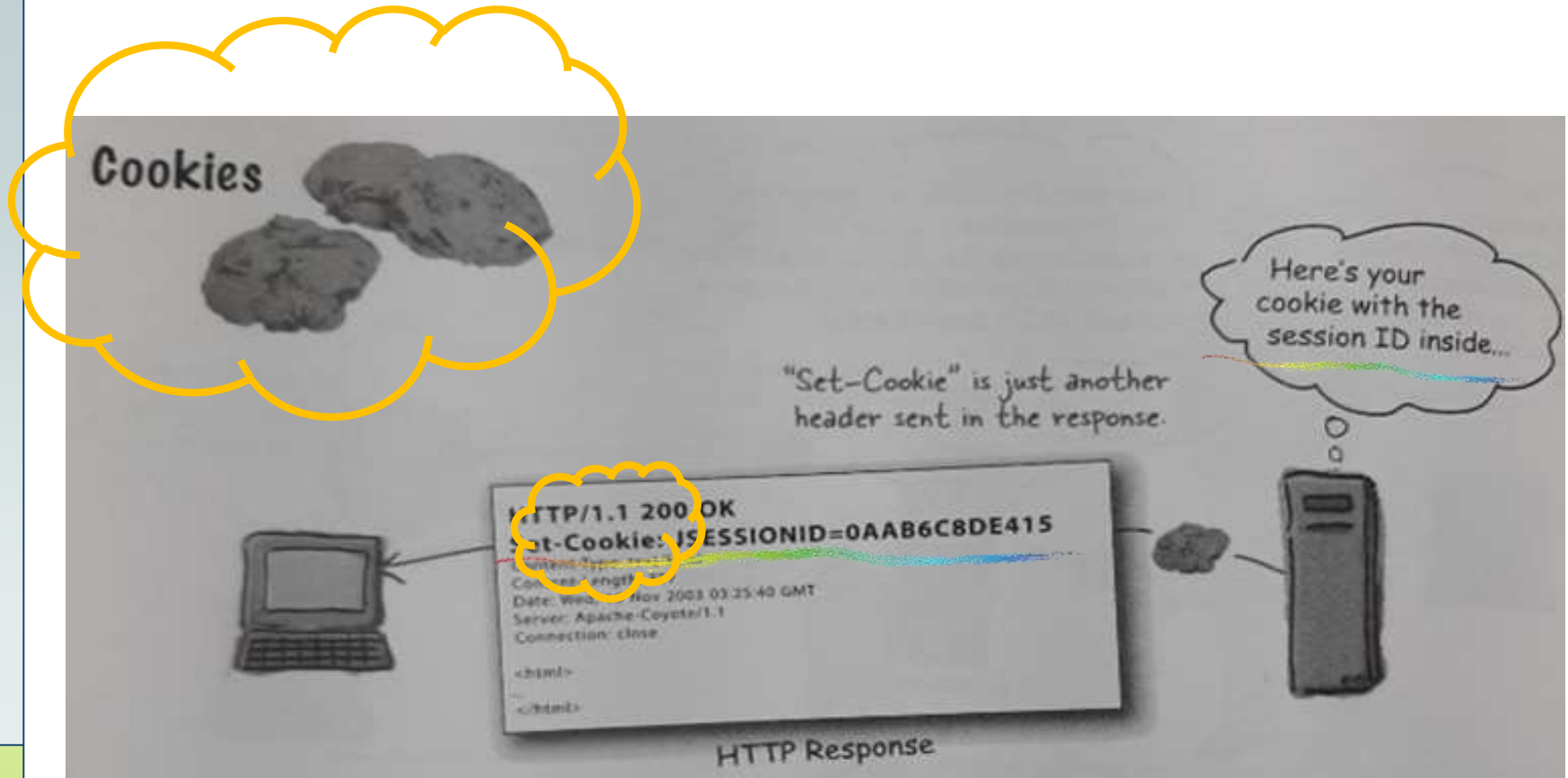
당연하지! 하지만, 난 기억력이 나빠서
널 기억하지 못할거야. 그러니, 너만을
위한 세션ID를 줄테니 다음 번에
request를 보낼 때마다 꼭 그 번호를
사용해! 그럼 너를 알아볼 수 있어~



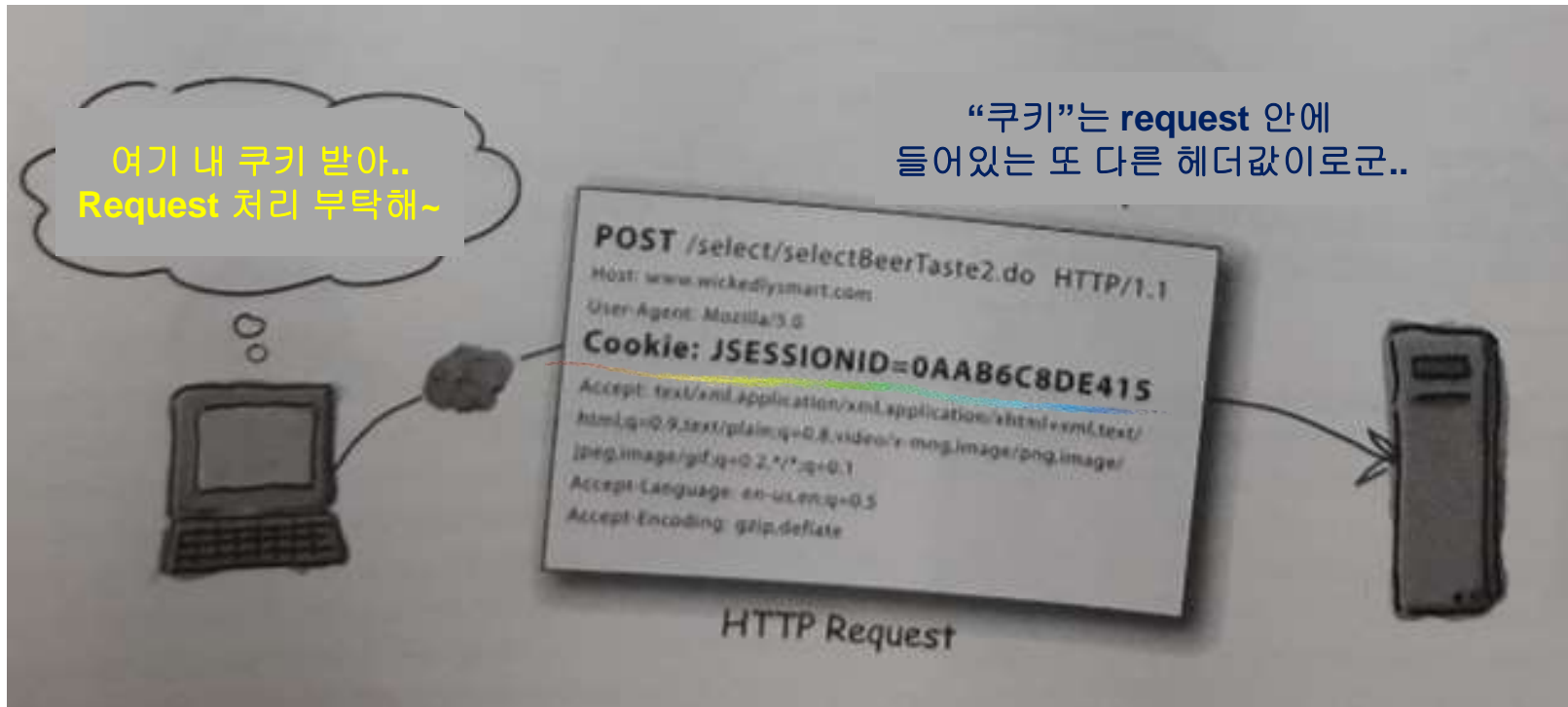
두 번째 이후 request도 처리 가능



고객과 컨테이너 간 세션ID 교환



후속 request에서 쿠키 사용





RESPONSE에서 세션 쿠키 보낼 때

HttpSession session = request.getSession();



고작 이게 전부임! 세션을 처리하기 위한 나머지 작업은 다 자동으로 됨

- 새로운 **HttpSession** 객체를 생성할 필요 없음
- 고유한 세션**ID** 만드는 것 안해도 됨
- 새로운 쿠키 객체 안 만들어도 됨
- 세션**ID**와 쿠키 연관성 찾는게 안해도 됨
- **response** 안에 쿠키 삽입하는 거 안해도 됨

여전히 컨테이너가 알아서 처리 중!



REQUEST로부터 세션ID 받을 때

```
HttpSession session = request.getSession();
```



IF (REQUEST에 세션ID 쿠키가 들어있음)

그 ID에 해당하는 세션을 찾는다!

ELSE IF (세션ID 쿠키가 없음 OR 그 ID에 맞는 활성 세션 없음)

새로 세션을 만든다!



❖ 필요성

- HTTP의 비연결성을 보완
- 장바구니와 같이 여러 페이지로 이동하더라도 사용자 정보와 필요 정보 유지 필요

❖ 쿠키

- 사용자 컴퓨터에 저장
 - 저장된 정보를 다른 사람 또는 시스템이 볼 수 있는 단점
- 유효시간이 지나면 사라짐

❖ 세션

- 서버의 메모리에 저장
- 서버가 종료되거나 유효시간이 지나면 사라짐



❖ 정의

- 쿠키는 서버에서 만들어진 작은 정보의 단위

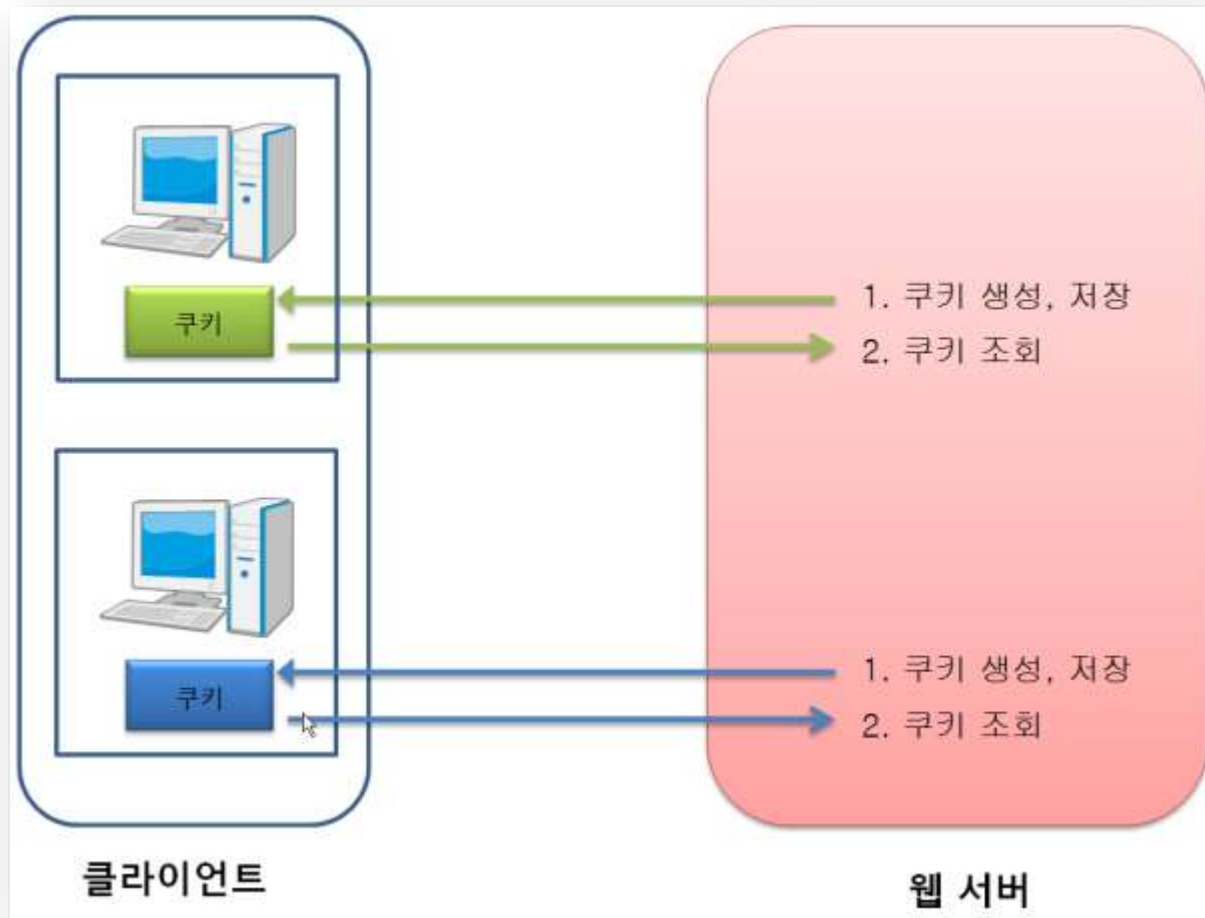
❖ 이용 방법

- 서버에서 클라이언트의 브라우저로 전송되어 사용자의 컴퓨터에 저장
- 저장된 쿠키는 다시 해당하는 웹 페이지에 접속 할 때, 브라우저에서 서버로 쿠키를 전송
- 쿠키는 이름(name)과 값(value)으로 구성된 자료를 저장
 - 이름과 값 외에도 주석(comment), 경로(path), 유효기간(maxage, expiry), 버전(version), 도메인(domain)과 같은 추가적인 정보를 저장

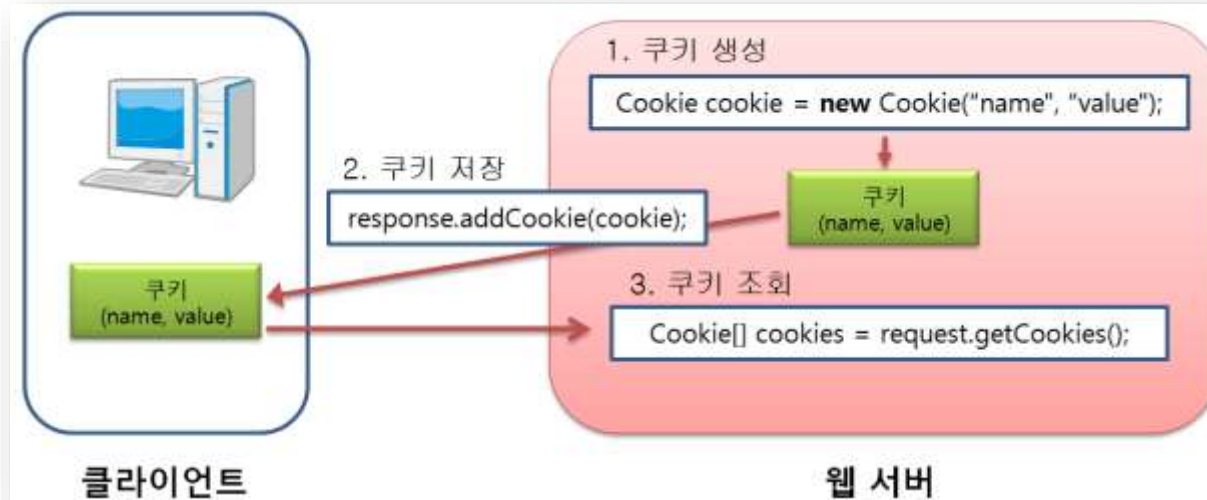
❖ 쿠키는 그 수와 크기에 제한

- 하나의 쿠키는 4K Byte 크기로 제한
- 브라우저는 각각의 웹사이트 당 20개의 쿠키를 허용
- 모든 웹 사이트를 합쳐 최대 300개를 허용
- 그러므로 클라이언트 당 쿠키의 최대 용량은 1.2M Byte

쿠키의 이용



❖ javax.servlet.http.Cookie 클래스



반환형

메소드 이름

메소드 기능

int **getMaxAge()**

쿠키의 최대지속 시간을 초단위로 지정
-1 일 경우 브라우저가 종료되면 쿠키를 만료

String **getName()**

쿠키의 이름을 스트링으로 반환

String **getValue()**

쿠키의 값을 스트링으로 반환

void **setMaxAge(int expiry)**

쿠키의 만료시간을 초단위로 설정

void **setValue(String newValue)**

쿠키에 새로운 값을 설정할 때 사용



❖ 쿠키 생성

- 쿠키는 (이름, 값)의 쌍 정보를 입력하여 생성
- 쿠키의 이름은 알파벳과 숫자로만 구성되고, 쿠키 값은 공백, 괄호, 등호, 콤마, 콜론, 세미콜론 등은 포함 불가능
 - `Cookie cookie = new Cookie("user", "kang");`

❖ 쿠키의 유효기간 설정

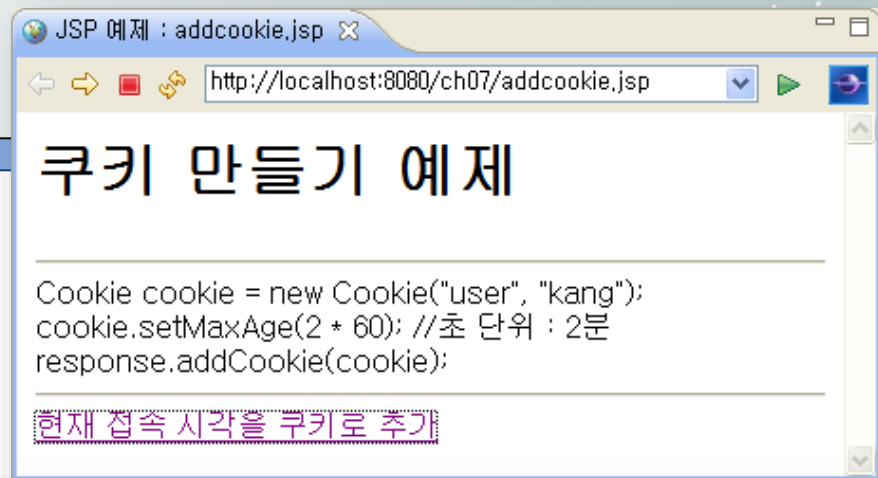
- 메소드 `setMaxAge()`
 - 인자는 유효기간을 나타내는 초 단위의 정수형
 - 만일 유효기간을 0으로 지정하면 쿠키의 삭제
 - 음수를 지정하면 브라우저가 종료될 때 쿠키가 삭제
- 유효기간을 2분으로 지정하려면
 - `cookie.setMaxAge(2 * 60);` //초 단위 : 2분
 - 1주일로 지정하려면 `(7*24*60*60)`로

❖ 클라이언트의 컴퓨터에 파일 형태로 저장

- 내장객체 `response`의 `addCookie` 메소드를 이용
 - `response.addCookie(cookie);`

쿠키 추가 예제(1)

❖ 예제 addcookie.jsp



쿠키 추가 예제(2)

❖ 예제 addtimecookie.jsp

현재 시각을 쿠키로 저장

```
String now = new java.util.Date().toString();  
Cookie cookie = new Cookie("lastconnect", now);  
cookie.setMaxAge(10); //초 단위 : 10초  
response.addCookie(cookie);
```

[쿠키 조회](#)

addtimecookie.jsp X JSP 예제 : cookie.jsp addcookie.jsp

```
1<%@ page language="java" contentType="text/html; charset=EUC-KR" pageEncoding="EUC-KR"%>  
2<html>  
3<head>  
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">  
5<title>JSP 예제 : addtimecookie.jsp</title>  
6</head>  
7<body>  
8    <h1>현재 시각을 쿠키로 저장</h1>  
9    <hr>  
10    String now = new java.util.Date().toString(); <br>  
11    Cookie cookie = new Cookie("lastconnect", now); <br>  
12    cookie.setMaxAge(10); //초 단위 : 10초 <br>  
13    response.addCookie(cookie); <br>  
14    <%  
15        String now = new java.util.Date().toString();  
16        Cookie cookie = new Cookie("lastconnect", now);  
17        cookie.setMaxAge(10); //초 단위 : 10초  
18        response.addCookie(cookie);  
19    %>  
20    <hr><a href=getcookie.jsp >쿠키 조회</a>  
21</body>  
22</html>
```



❖ 클라이언트에 저장된 쿠키를 조회

- 내장객체 request의 get_cookies() 메소드를 이용
- 메소드 get_cookies()의 반환 값은 저장된 모든 쿠키의 배열
 - 쿠키가 없으면 null 값이 반환

```
Cookie[] cookies = request.getCookies();
```

❖ 각각의 쿠키를 얻는 방법

- 쿠키 배열 변수 cookies는 다음과 같이 for each 문 이용
 - Cookie의 getName(), getValue() 메소드를 이용

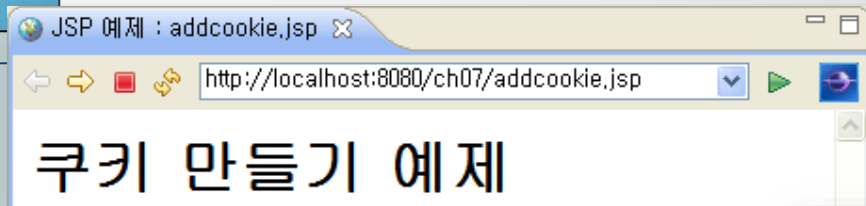
```
for (Cookie c : cookies) {  
    out.println("쿠키 이름(name) : " + c.getName() + ", " );  
    out.println("쿠키 값(value) : " + c.getValue() + "<br>" );  
}
```

쿠키 조회 예제

❖ 예제 getcookies.jsp

```
getcookies.jsp
1<%@ page language="java" contentType="text/html; charset=EUC-KR" pageEncoding="EUC-KR"%>
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>JSP 예제 : getcookies.jsp</title>
6</head>
7<body>
8    <h1>쿠키 조회 예제</h1>
9    <hr>
10    <%
11        Cookie[] cookies = request.getCookies();
12        if (cookies == null) {
13            out.println("쿠키가 없습니다.");
14        } else {
15            /*
16            for (int i=0; i<cookies.length; ++i) {
17                out.println("쿠키 이름 (name) : " + cookies[i].getName() + ", " );
18                out.println("쿠키 값 (value) : " + cookies[i].getValue() + "<br>" );
19            }
20            */
21            for (Cookie c : cookies) {
22                out.println("쿠키 이름 (name) : " + c.getName() + ", " );
23                out.println("쿠키 값 (value) : " + c.getValue() + "<br>" );
24            }
25        }
26    %>
27</body>
28</html>
```


쿠키 예제 결과



```
Cookie cookie = new Cookie("user", "kang");  
cookie.setMaxAge(2 * 60); //초 단위 : 2분  
response.addCookie(cookie);
```

현재 접속 시각을 쿠키로 추가

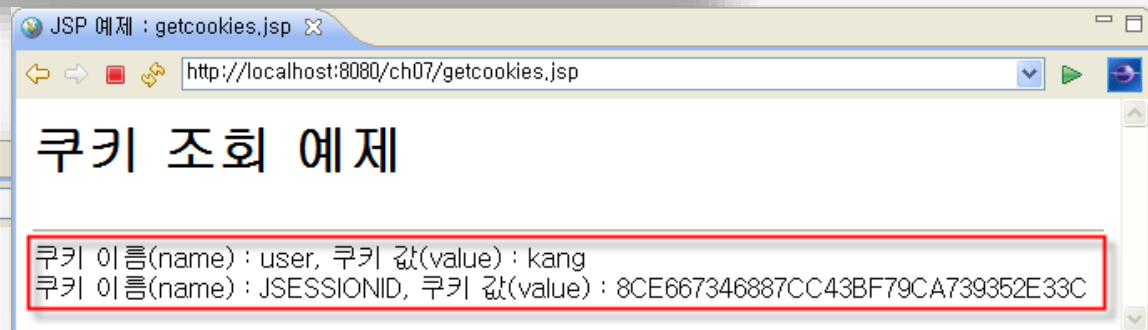
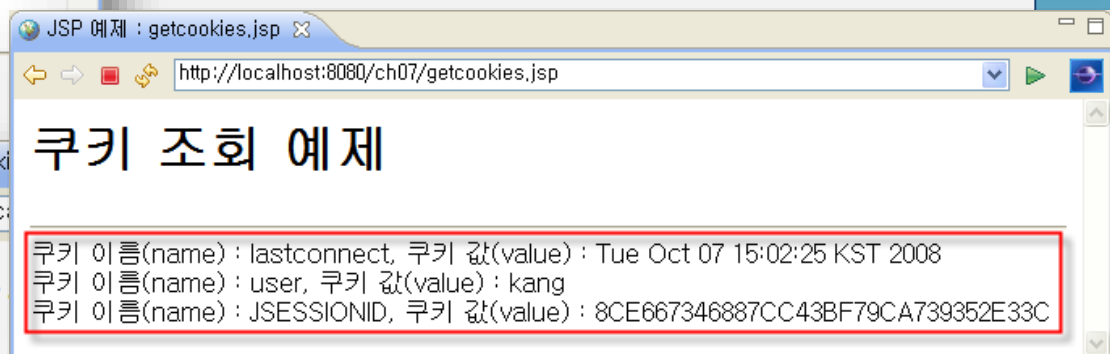
JSP 예제 : addtimecook

http://loc

현재 시각

```
String now = new java.util.Date().toString();  
Cookie cookie = new Cookie("lastconnect", now);  
cookie.setMaxAge(10); //초 단위 : 10초  
response.addCookie(cookie);
```

쿠키 조회



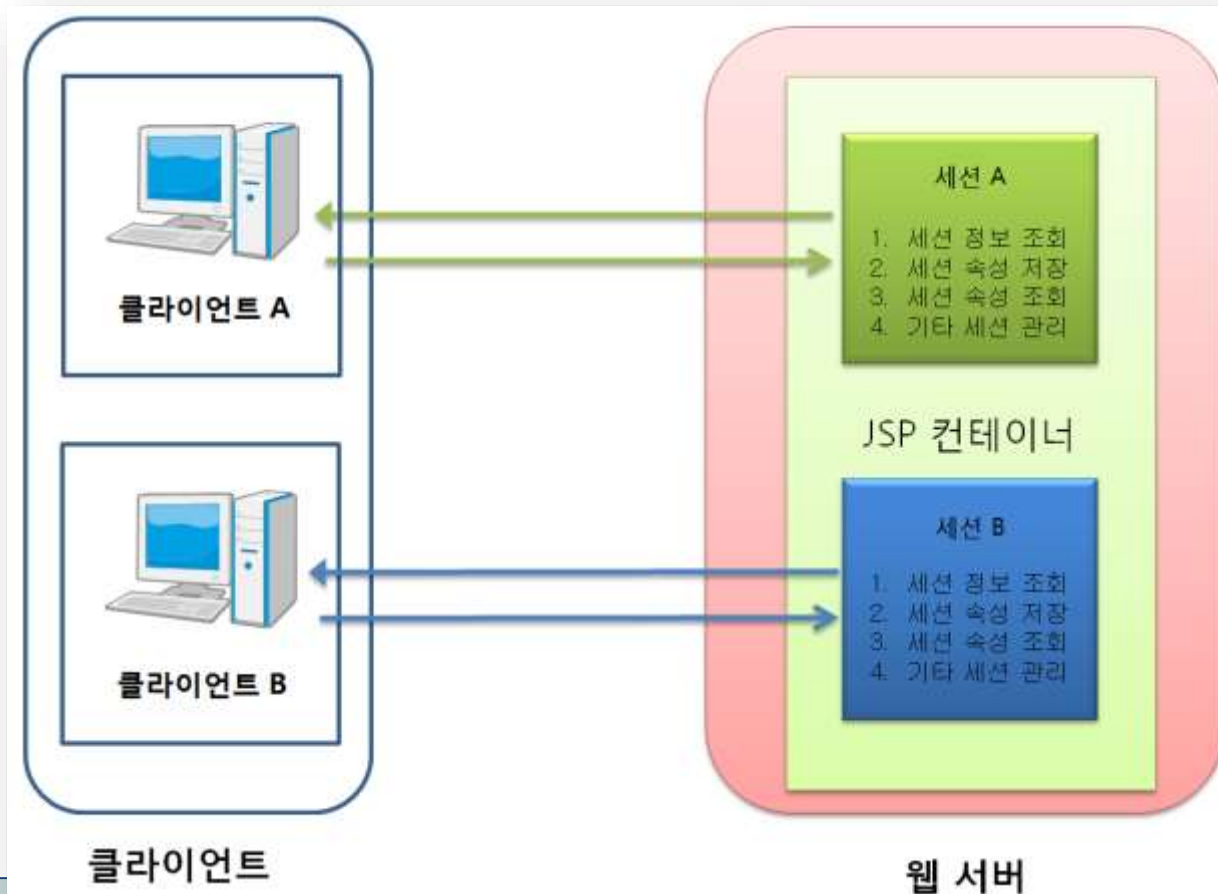
쿠키 조회 예제

쿠키 이름(name) : JSESSIONID, 쿠키 값(value) : 8CE667346887CC43BF79CA739352E33C



❖ 개념

- 클라이언트의 정보를 클라이언트 PC에 저장하는 것이 쿠키
- 클라이언트 마다 각기 다른 정보를 서버에 저장하는 것이 세션
 - 즉 세션은 클라이언트 사용자 별로 여러 페이지 이동을 인식
 - 필요한 정보를 서버에 저장, 조회 방법과 세션을 관리할 수 있는 방법을 제공





❖ 패키지 javax.servlet.http

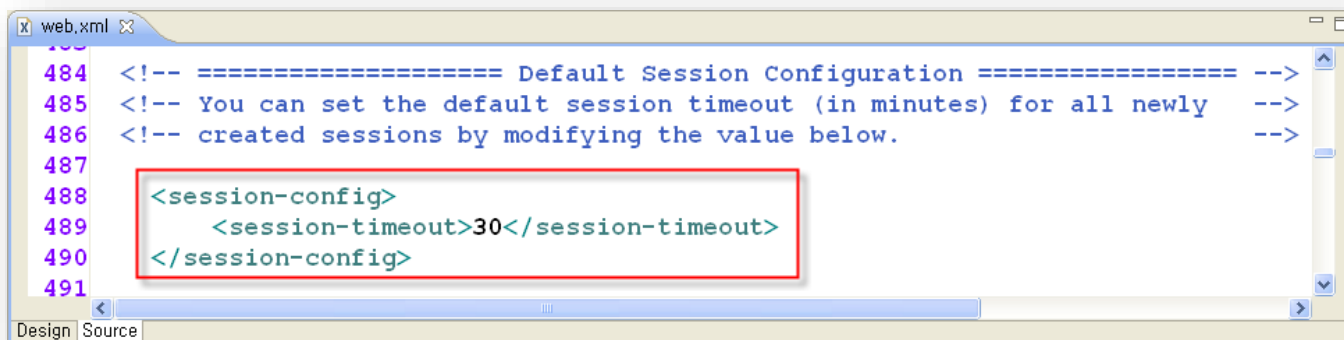
■ 인터페이스 HttpSession

반환형	메소드 이름	메소드 기능
long	<u>getCreationTime()</u>	를 기준으로 하여 현재 세션이 생성된 시간까지 지난 시간을 계산하여 밀리세컨드로 반환
String	<u>getId()</u>	세션에 할당된 유일한 식별자(ID)를 String 타입으로 반환
int	<u>getMaxInactiveInterval()</u>	현재 생성된 세션을 유지하기 위해 설정된 최대 시간을 초의 정수 형으로 반환, 지정하지 않으면 기본 값은 1800초, 즉 30분이며, 기본 값도 서버에서 설정 가능

반환형	메소드 이름	메소드 기능
Object	<u>getAttribute(String name)</u>	name이란 이름에 해당되는 속성값을 Object 타입으로 반환, 해당되는 이름이 없을 경우에는 null을 반환
Enumeration	<u>getAttributeNames()</u>	속성의 이름들을 Enumeration 타입으로 반환
void	<u>invalidate()</u>	현재 생성된 세션을 무효화 시킴
void	<u>removeAttribute(String name)</u>	name으로 지정한 속성의 값을 제거
void	<u>setAttribute(String name, Object value)</u>	name으로 지정한 이름에 value 값을 할당
void	<u>setMaxInactiveInterval(int interval)</u>	세션의 최대 유지시간을 초 단위로 설정
boolean	<u>isNew()</u>	세션이 새로이 만들어졌으면 true, 이미 만들어진 세션이면 false를 반환

❖ 세션은 클라이언트가 서버에 접속하는 순간 생성

- 특별히 지정하지 않으면 세션의 유지 시간은 기본 값으로 30분 설정
- 세션의 유지 시간이란 서버에 접속한 후 서버에 요청을 하지 않는 최대 시간
- 30분 이상 서버에 전혀 반응을 보이지 않으면 세션이 자동으로 끊어짐.
- 이 세션 유지 시간은 서버에서 설정 가능
 - 톰캣에서 설치 폴더 하부 [conf] 폴더에 파일 web.xml
 - session timeout으로 30초가 지정되어 있는 것을 확인



The screenshot shows a text editor window titled 'web.xml'. The content is XML code for session configuration. A red rectangle highlights the following code block:

```
488 <session-config>
489   <session-timeout>30</session-timeout>
490 </session-config>
```

The rest of the visible code includes comments and the opening tag for the session configuration.

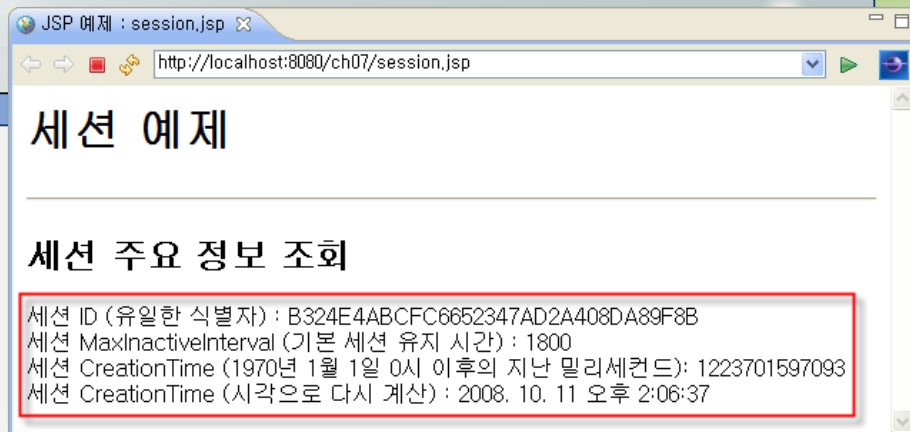


❖ 내장객체 session 메소드 *getCreationTime()*

- 현재 세션이 생성된 시간을 1970년 1월 1일 0시를 기준으로 지난 시간을 계산하여 밀리세컨드로 반환
- 그러므로 이를 년, 월, 일 시의 시간정보로 출력하려면 클래스 `java.util.Date`의 생성자 `Date(long mseconds)`를 이용하여 객체를 만든 후 출력
- `long mseconds = session.getCreationTime();`
- `Date time = new Date(mseconds);`

주요 세션 정보 조회

❖ 예제 session.jsp





❖ 내장객체 session 메소드 `setAttribute(name, value)`

- `setAttribute(String name, Object value)`
- `name`과 `value`의 쌍으로 객체 `Object`를 저장하는 메소드
- 세션이 유지되는 동안 저장할 자료를 저장
- `session.setAttribute("id", "javajsp");`
- `session.setAttribute("pwd", "jdktomcat");`

❖ `getAttribute(String name)` 메소드

- 세션에 저장된 자료는 다시 `getAttribute(String name)` 메소드를 이용해 조회
- 반환 값은 `Object` 유형이므로 저장된 객체로 자료유형 변환이 필요
- 메소드 `setAttribute()`에 이용한 `name`인 “id”를 알고 있다면 바로 다음과 같이 바로 조회
- `String value = (String) session.getAttribute("id");`



❖ 메소드 *getAttributeNames()*

- 세션의 속성으로 지정한 이름을 모두 알기 위해서
- 반환 값은 인터페이스 Enumeration
 - 패키지 *java.util*
 - 페이지 지시자의 *import* 속성을 이용
- `<%@ page import="java.util.Enumeration, java.util.Date" %>`
- ...
- `Enumeration<String> e = session.getAttributeNames();`

❖ *Enumeration<String>*

- 자료유형 *Enumeration*은 여러 개의 내부 원소를 일렬로 저장한 구조를 지원
- 객체 *e*의 자료유형은 *Enumeration<String>*
 - *<String>*의 일반화 유형으로 선언
 - *e.nextElement()*의 반환 값을 저장할 때 *String*으로 자료유형 변환이 필요 없는 장점



```
❖ Enumeration<String> e = session.getAttributeNames();  
  
❖ while ( e.hasMoreElements() ) {  
❖     String name = e.nextElement();  
❖     String value = (String) session.getAttribute(name);  
❖     out.println("세션 name : " + name + ", ");  
❖     out.println("세션 value : " + value + "<br>");  
❖ }
```

반환형	메소드 이름	메소드 기능
boolean	hasMoreElements()	enumeration의 내부에 더 이상의 원소가 있는지 결과를 반환, 있다면 true, 없으면 false를 반환
Object	nextElement()	enumeration의 내부에 더 이상의 원소가 있다면 다음 원소를 반환

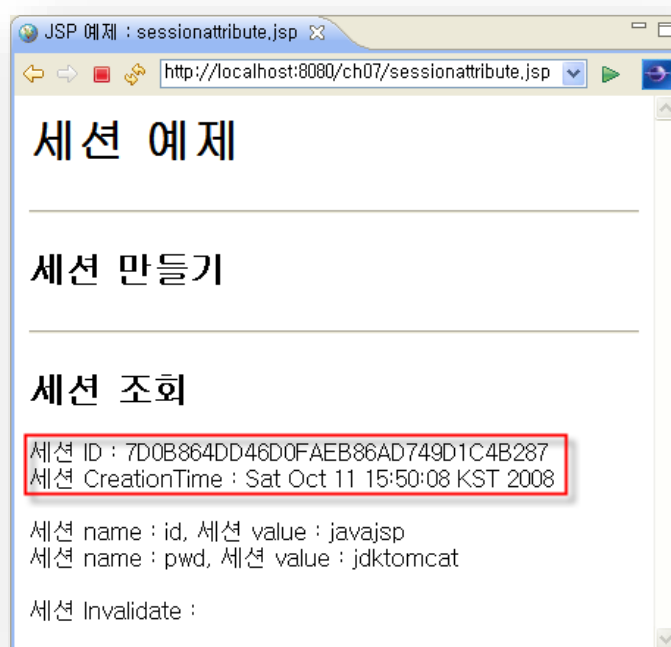
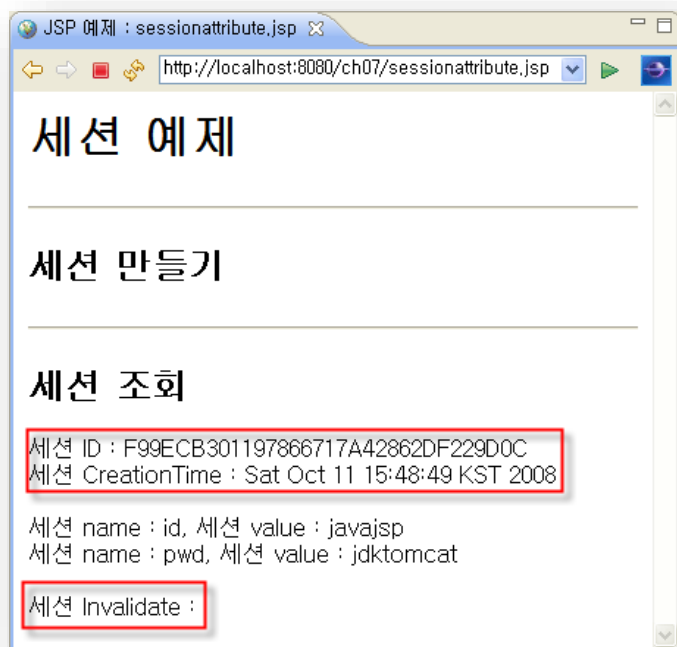
❖ 예제 sessionattribute.jsp

```
JSP 예제 : sessionattribute.jsp sessionattribute.jsp
1<%@ page language="java" contentType="text/html; charset=EUC-KR" pageEncoding="EUC-KR"%>
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>JSP 예제 : sessionattribute.jsp</title>
6</head>
7<body>
8    <%@ page import="java.util.Enumeration, java.util.Date" %>
9    <h1>세션 예제</h1>
10   <hr><h2>세션 만들기</h2>
11   <%
12       session.setAttribute("id", "javajsp");
13       session.setAttribute("pwd", "jdktomcat");
14   %>
15   <hr><h2>세션 조회</h2>
16   세션 ID : <%= session.getId() %><br>
17   세션 CreationTime : <%= new Date(session.getCreationTime()) %><br><br>
18   <%
19       Enumeration<String> e = session.getAttributeNames();
20       while ( e.hasMoreElements() ) {
21           String name = e.nextElement();
22           String value = (String) session.getAttribute(name);
23           out.println("세션 name : " + name + ", ");
24           out.println("세션 value : " + value + "<br>");
25       }
26   %>
27   <br><br>세션 Invalidate : <%= session.invalidate(); %>
28</body>
29</html>
```

❖ 예제 sessionattribute.jsp

❖ 마지막 부분에서

- 다음과 같이 session.invalidate()를 호출
 -
세션 Invalidate : <% session.invalidate(); %>
 - 이전 세션을 무조건 무효화시킴
- 다시 페이지를 refresh하면 항상 세션 ID와 생성시간이 바뀜





- ❖ 내장객체인 session의 메소드 `setAttribute()`를 이용해 세션의 ID와 생성시간을 저장
 - `session.setAttribute("id", session.getId());`
 - `session.setAttribute("time", new Date(session.getCreationTime()));`
- ❖ 5초 이상 서버에 반응을 하지 않으면 세션을 무효화 시킴
 - `session.setMaxInactiveInterval(5);`
- ❖ 이미 지정된 세션 속성을 삭제
 - 메소드 `removeAttribute("id")`와 같이 속성 이름을 인자로 호출
 - `session.removeAttribute("id");`
- ❖ 메소드 `isNew()`
 - 세션이 새로 만들어진 것인지를 `boolean` 유형으로 반환

세션이 생성된 이후 시간, 무반응 시간 계산

❖ 현재 세션이 만들어진 이후 지난 시간을 출력

- 클래스 Date의 메소드 getTime()
 - Date의 시간정보를 1970년 1월 1일 0시 이후의 밀리세컨드 초로 반환
-
- long nowtime = new Date().getTime();
- <% long sessiontime = nowtime - session.getCreationTime(); %>
- 세션이 만들어진 이후 지난 시간 : <%=sessiontime/1000 %>초
-

❖ 현재 페이지에서 서버에 반응을 보이지 않은 시간 계산

- 현재 시간인 nowtime에서 이전에 참조한 시간이 저장된 beforetime를 뺀
- beforetime은 이전에 참조한 시간을 저장한 변수이므로 소속 변수로 선언
 - 페이지를 종료할 때 다시 nowtime을 beforetime에 저장
-
- <%! long beforetime = new Date().getTime();
- //이전 페이지 참조 시간을 저장하는 소속 변수 %>
-
- <% long inactiveinterval = nowtime - beforetime; %>
- 서버에 반응을 보이지 않은 시간 : <%=inactiveinterval/1000 %>초
-
- <% beforetime = nowtime; %>

세션 시간 예제 sessiontimeout.jsp

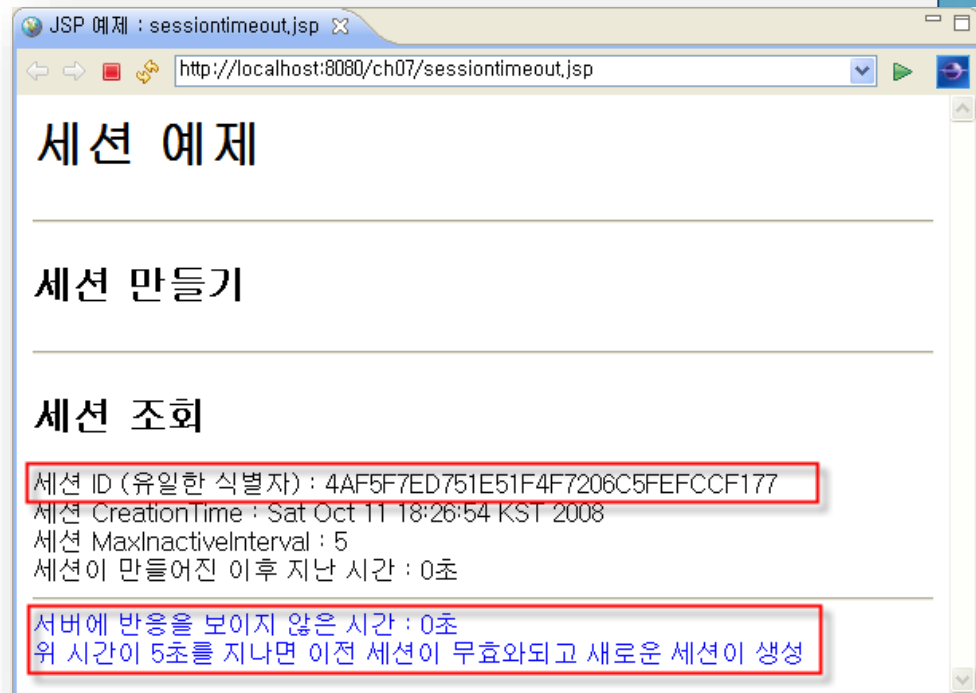
```
JSP 예제 : sessiontimeout.jsp sessiontimeout.jsp
1<%@ page language="java" contentType="text/html; charset=EUC-KR" pageEncoding="EUC-KR"%>
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>JSP 예제 : sessiontimeout.jsp</title>
6</head>
7<body>
8    <%@ page import="java.util.Enumeration, java.util.Date" %>
9    <h1>세션 예제</h1>
10   <hr><h2>세션 만들기</h2>
11   <%! long beforetime = new Date().getTime(); //이전 페이지 참조 시간을 저장하는 소속 변수 %>
12   <%
13       long nowtime = new Date().getTime();
14       if ( session.isNew() ) {
15           session.setAttribute("id", session.getId());
16           session.setAttribute("time", new Date(session.getCreationTime()));
17           session.setMaxInactiveInterval(5);
18       } else {
19           session.removeAttribute("id");
20       }
21   %>
22   <hr><h2>세션 조회</h2>
23   세션 ID (유일한 식별자) : <%= session.getAttribute("id") %><br>
24   세션 CreationTime : <%=session.getAttribute("time") %><br>
25   세션 MaxInactiveInterval : <%=session.getMaxInactiveInterval() %><br>
26   <% long sessiontime = nowtime - session.getCreationTime(); %>
27   세션이 만들어진 이후 지난 시간 : <%=sessiontime/1000 %>초
28
29   <font color=blue><hr>
30   <% long inactiveinterval = nowtime - beforetime; %>
31   서버에 반응을 보이지 않은 시간 : <%=inactiveinterval/1000 %>초<br>
32   위 시간이 <%=session.getMaxInactiveInterval() %>초를 지나면
33   이전 세션이 무효화되고 새로운 세션이 생성</font><br>
34
35   <% beforetime = nowtime; %>
36</body>
37</html>
```

❖ 5초 이전 refresh

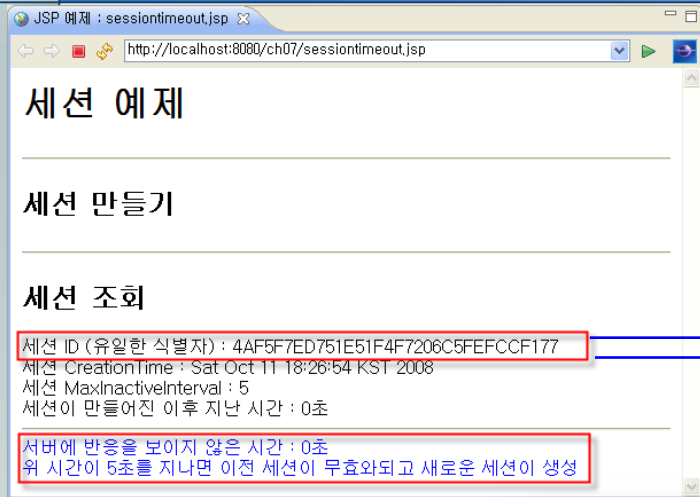
- 세션 유효시간으로 설정한 5초 이전에 다시 페이지를 refresh하면 동일한 세션이므로 세션 ID가 삭제되어 null이 출력

❖ 5초 이후 refresh

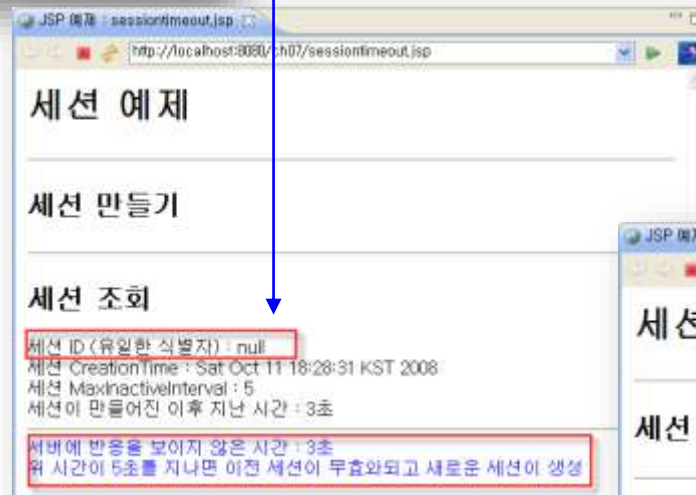
- 만일 세션 유효시간으로 설정한 5초가 지난 후 페이지를 refresh하면 세션이 무효화
- 다시 새로운 세션이 설정되므로 조회되는 세션 ID가 이전 것과 다름



예제 sessiontimeout.jsp refresh 결과

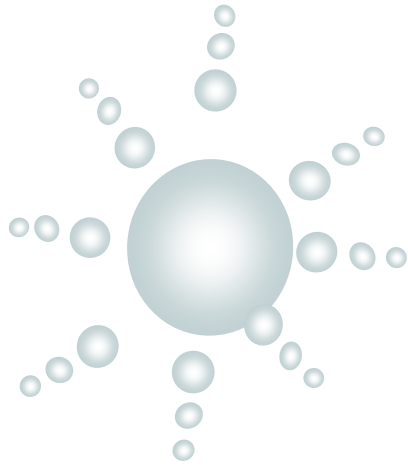


⑩ 세션 유효시간으로 설정한 5초 이전에 다시 페이지를 refresh하면 동일한 세션이므로 세션 ID가 삭제되어 null이 출력되는 것을 볼 수 있다.



⑩ 만일 세션 유효시간으로 설정한 5초가 지난 후 페이지를 refresh하면 세션이 무효화되어 다시 새로운 세션이 설정되므로 조회되는 세션 ID가 이전 것과 다른 것을 볼 수 있다.





Thank You !